Research Article https://doi.org/10.12973/eu-jer.14.4.1137

European Journal of Educational Research
Volume 14, Issue 4, 1137 - 1149.
ISSN: 2165-8714
http://www.eu-jer.com/

Computational Thinking Through Scaffolded Game Development Activities: A Study with Graphical Programming

Nurul Hazlina Noordin*
Universiti Malaysia Pahang Al-Sultan Abdullah, MALAYSIA

Received: March 2, 2025 ▪ Revised: May 30, 2025 ▪ Accepted: July 5, 2025

Abstract: This study investigates the effectiveness of scaffolded game development activities in enhancing computational thinking (CT) skills among young learners using a graphical programming environment. While prior research highlights the value of block-based programming in CT education, few studies explore how structured scaffolding supports learners in completing full game projects. Grounded in Vygotsky's Zone of Proximal Development and Wing's CT framework, this study involved 310 participants aged 10 to 15, including their teachers, in a tiered sequence of programming tasks using mBlock programming platform. Learners progressed from basic to more complex programming constructs, namely, loops, conditionals, variables, and debugging, which are included in the development of a complete Pac-Man or Snake game. Quantitative results demonstrated significant improvements in CT skills across all age groups. Qualitative data revealed increased learner engagement, reduced programming anxiety, and enhanced interest in computational problem-solving. The findings suggest that scaffolded game development is a promising strategy for early CT instruction, offering both cognitive and affective benefits. This work contributes to current literature by demonstrating how structured support and creative programming tasks can jointly promote CT proficiency and learner motivation in foundational computing education.

Keywords: *Computational thinking, game development, graphical programming, tiered scaffolding.*

To cite this article: Noordin, N. H. (2025). Computational thinking through scaffolded game development activities: A study with graphical programming. *European Journal of Educational Research*, *14*(4), 1137-1149. https://doi.org/10.12973/eu-jer.14.4.1137

Introduction

Computational thinking (CT) has emerged as a critical skill for the 21st century, essential for solving complex problems across various disciplines. As technology continues to evolve, CT is becoming increasingly important, not only for computer scientists but for learners of all backgrounds (Bachiller-Burgos et al., 2020). This study explores the integration of scaffolded game development activities into early education, leveraging graphical programming environments to enhance CT skills.

One such approach is the use of graphical programming languages, such as Scratch, Blockly, and mBlock, which provide intuitive, visual interfaces that reduce the entry barrier for novice programmers (Dilmen et al., 2023; Sun et al., 2024). These environments enable learners to create interactive games, animations, and digital stories, transforming programming from a text-heavy task into an exploratory, creative process (Margolis, 2020; Noordin et al., 2024). Game development has been shown to improve student engagement and motivation, while providing a meaningful context in which CT concepts can be applied.

Another critical factor in CT education is the role of scaffolding. Drawing from Vygotsky's Zone of Proximal Development (ZPD), scaffolding refers to the temporary instructional support that enables learners to complete tasks they could not accomplish independently (van de Pol et al., 2015). In programming contexts, scaffolding may include structured tutorials, sequenced challenges, and peer or mentor guidance, all of which help learners build skills gradually and with confidence. Although previous studies highlight the role of scaffolding in STEM education, limited research has explored its use specifically in game development tasks within graphical programming environments.

This study addresses that gap by evaluating the use of scaffolded game development activities to enhance CT skills among young learners. Specifically, the intervention used a tiered scaffolding approach within the mBlock environment, where

* **Correspondence:**
Nurul Hazlina Noordin, UMPSA STEM Lab, Universiti Malaysia Pahang Al-Sultan Abdullah, Pekan Pahang, Malaysia. ✉ hazlina@umpsa.edu.my

students aged 10 to 15 completed a sequence of activities culminating in the creation of a complete Pac-Man or Snake game.

Research Questions

The study aims to assess the effectiveness of this approach in supporting CT skill development and to explore learners' perceptions of their programming experiences. It is guided by the following research questions:

1. To what extent do scaffolded game development activities improve CT skills among young learners in a graphical programming environment?
2. How do students perceive the scaffolded game development experience in terms of engagement, confidence, and interest in programming?

Literature Review

Block-Based Programming and CT Development

Block-based programming environments such as Scratch, Blockly, and mBlock have become popular tools in introductory programming education due to their intuitive, visual interfaces. These platforms reduce the cognitive load associated with syntax, enabling learners to focus on core problem-solving processes (Rao & Bhagat, 2024). Numerous studies confirm the effectiveness of these tools in promoting CT skills, including decomposition, pattern recognition, abstraction, and algorithmic thinking (Srisangngam & Dechsura, 2020; Wing, 2006).

Recent work by Dilmen et al. (Dilmen et al., 2023) demonstrated how block-based platforms like Code.org enhanced learners' algorithmic thinking through accessible interfaces and engaging tasks. Similarly, Sun et al. (Sun et al., 2024) found that students using block-based environments demonstrated more frequent debugging behaviors and deeper CT engagement compared to those using text-based languages. These findings reinforce the pedagogical value of graphical tools for novice learners. However, most of these studies evaluate either discrete tasks or brief modules focused on individual CT components. There is limited research that explores how block-based programming can be scaffolded into larger, outcome-based projects, such as complete games, which more closely mirror authentic software development processes. Additionally, few studies examine how tiered support over time affects learners' progression from guided exercises to independent creation. Extending beyond programming, Ye et al. (2023) demonstrated that computational thinking also enhances mathematical reasoning in K-12 settings, reinforcing the cross-disciplinary value of CT skills developed through iterative and problem-based learning approaches, which can be effectively supported by block-based tools (Ye et al., 2023).

Scaffolding in Programming Education

Scaffolding, based on Vygotsky's concept of the Zone of Proximal Development (ZPD), refers to the structured support provided to learners to help them accomplish tasks just beyond their current capability (Shabani et al., 2010). In programming education, scaffolding may take the form of worked examples, incremental challenges, mentor support, or peer collaboration—each enabling learners to build confidence and competence over time (van de Pol et al., 2015).

Research by Fernández et al. (Fernández et al., 2001) redefined scaffolding as a dynamic, dialogic process that can occur not only between teacher and student but also among peers. More recently, Shin et al. (2025) demonstrated that the use of cognitive and metacognitive scaffolds in collaborative programming tasks reduced cognitive load and enhanced problem-solving ability (Shin et al., 2025). Wan et al. (2024) compared structuring-oriented and problematization-oriented scaffolding in video-based programming instruction and found that both approaches significantly improved students' CT skills, especially in algorithmic thinking (Wan et al., 2024).

Despite this growing body of evidence, there remains a gap in studies that examine how scaffolding can be applied progressively and systematically to support the completion of full digital products, such as playable games, within block-based environments. This is especially relevant for young learners, who may require ongoing, adaptive support to transition from isolated exercises to complex, multi-stage tasks.

Computational Thinking as a Framework

Wing emphasized CT as a core cognitive skill involving decomposition, pattern recognition, abstraction, and algorithmic thinking skills that are not only central to computer science but broadly applicable across disciplines (Wing, 2006). CT has since been recognized as essential for preparing learners to navigate and solve problems in increasingly digital and data-rich environments.

Lodi and Martini (Lodi & Martini, 2021) traced the conceptual development of CT, drawing on Papert's constructionist ideas and Wing's advocacy for early integration in education. Their work underscores the value of CT as a cross-disciplinary, foundational literacy, especially when taught through interactive and constructivist methods. However, they

also noted that the design of CT curricula often lacks concrete pathways for students to meaningfully apply these skills in extended, authentic tasks.

Integrating CT and Scaffolding through Game Development

Combining scaffolding with game-based programming provides a promising model for CT instruction. Games naturally lend themselves to CT practices, such as breaking down problems, identifying patterns, designing logic, and debugging behavior. When paired with a scaffolded instructional model, game development becomes an opportunity not just for skill acquisition, but also for creative problem-solving and sustained engagement. While numerous studies have explored CT skill acquisition or motivational outcomes, these aspects are often addressed in isolation. Only a limited number of investigations have systematically explored how tiered scaffolding supports learners in independently completing a full game, an approach that demonstrates CT competencies in a more integrated and meaningful context. Additionally, affective dimensions of learning, such as confidence, anxiety, and interest, are seldom examined in tandem with cognitive outcomes.

Research Focus

This study addresses these gaps by evaluating a structured, scaffolded intervention using mBlock, where learners progressed through a tiered sequence of programming challenges that culminated in the creation of complete Pac-Man or Snake games. The intervention was designed to integrate CT concepts within a meaningful, end-to-end development process supported by graduated instructional scaffolds. By investigating both quantitative skill gains and qualitative perceptions, this study offers new insights into how CT instruction can be made both effective and engaging for young learners through scaffolded, game-based programming.

Methodology

Research Design

This study employed a mixed-methods design, combining quantitative assessments with qualitative feedback to evaluate the effectiveness of scaffolded game development activities in enhancing CT skills. A pre-test and post-test model was used to measure learning gains, complemented by semi-structured interviews to capture students' perceptions of the learning experience.

Participants

The participants comprised 310 individuals, including students aged 10 to 15 years and their teachers, with a gender distribution of 113 females and 197 males, recruited from 12 schools across Malaysia. A convenience sampling method was used in collaboration with participating schools in the program. Participants were selected to ensure diversity in prior programming exposure, ranging from no experience to beginner familiarity with graphical programming environments. This diversity aimed to provide a comprehensive understanding of the scaffolding approach's influence across varying levels of novice proficiency.

The study was conducted with ethical approval from the IIUM Research Ethics Committee (IREC2023-173). Parental consent and student assent were obtained, and all participant data were anonymized to maintain confidentiality and comply with data protection standards.

Intervention and Learning Activities

Participants engaged in a series of scaffolded programming activities using mBlock, a graphical programming platform based on Scratch. The intervention was structured into tiered levels of difficulty, following Vygotsky's Zone of Proximal Development (ZPD) principles. These stages were specifically designed to move students from fully guided practice to independent problem-solving.

Both students and teachers took part as learners throughout the program, engaging in hands-on tasks designed to build their understanding of core computational thinking (CT) concepts. At the beginning of the program, participants received detailed step-by-step instruction and demonstrations on basic programming concepts, such as sprite movement, loops, conditionals, code debugging and sprite movement. These introductory tasks were delivered through instructor-led tutorials and worksheets, providing the initial support necessary for novice learners.

As learners became more comfortable, scaffolding was gradually reduced. The tasks increased in complexity, involving character interaction, score tracking with variables, and algorithmic logic to simulate full game functionality. Students were encouraged to decompose problems, design game mechanics, and troubleshoot errors independently or in small groups. This progression represents the *"fading"* of scaffolding, a core principle of ZPD, allowing students to operate just beyond their comfort zone with decreasing reliance on external help.

Instructors acted as facilitators, providing guidance, answering questions, and monitoring progress. They also encouraged peer-to-peer support by promoting collaborative problem-solving among students. Meanwhile, researchers observed the implementation, ensured consistency across sessions, and supported the teachers when necessary. They also collected observational data, documented engagement levels, and administered pre- and post-tests at designated points.

The outcome of the intervention was the development of a fully functional Pac-Man or Snake game, independently applying previously acquired skills. This final task required learners to synthesize concepts such as decomposition, abstraction, conditionals, and iterative logic, thereby showcasing mastery developed through scaffolded support.

Instruments and Data Collection

Quantitative data were collected through structured pre- and post-tests designed to assess students' computational thinking skills. These assessments evaluated core competencies such as decomposition, pattern recognition, abstraction, and algorithmic thinking within the context of graphical programming. The instruments were adapted from validated CT assessment frameworks used in K–12 computing education (Brennan & Resnick, 2012), and were piloted with 38 students to ensure clarity and content validity. Internal consistency was assessed using Cronbach's alpha ($\alpha = 0.92$), indicating a good reliability.

Qualitative data were gathered through semi-structured interviews and reflective surveys, exploring participants' experiences, challenges, and perceptions of the scaffolded game development activities. Participants were asked what they had experienced or learned during the program, their future ambitions, and their interest in programming or related fields. They were also invited to reflect on the support received, with prompts such as, *"Was the instructor/mentor helpful? Describe how."* Additional questions addressed the most challenging topics encountered, areas they wished to explore further, and whether they would recommend the program to peers, along with their reasoning. The question on instructor and mentor support was specifically intended to capture participants' perceptions of instructional scaffolding. In line with Vygotsky's Zone of Proximal Development (ZPD), this interview aimed to explore how structured guidance facilitated learners' progression from guided instruction to independent problem-solving. Example of such guidance include step-by-step explanations, real-time troubleshooting, and motivational encouragement.

The interviews were conducted after the intervention with a sample of 38 participants representing diverse ages and prior experience levels. Each interview lasted approximately 15–20 minutes and followed a predefined protocol aligned with the study's research questions.

Student perceptions were captured via a 7-item Likert-scale survey (A1–A7), which assessed aspects of programming understanding, skill development, and engagement. Perception data were collected as to complement quantitative learning outcomes with affective insights, and to understand how students experienced the scaffolded game development process, including their motivation, confidence, and engagement levels. This approach is supported by previous research that highlights the importance of affective dimensions, such as interest, enjoyment, and self-efficacy, in enhancing learning outcomes in computational thinking and programming education (Grover & Pea, 2013; Lye & Koh, 2014). Table 1 outlines the survey items, where highest-rated items were A2 and A6, reflecting increased confidence in physical computing and perceived technical skill development.

Table 1. Survey Items

Items	Details
A1	I understand better about Python Programming after attending the course
A2	I learn about the basics of physical computing after attending the course
A3	I am more interested in programming and physical computing after attending the course
A4	I am able to explore innovative solutions using Python Programming after attending the course
A5	I am confident to program in Python after attending the course
A6	This course has provided opportunities for me to improve my technical skills in preparation for my class projects
A7	I would recommend this course to my colleagues

Data Analysis

Pre- and post-test scores were analyzed using paired sample t-tests to determine statistically significant learning gains. Descriptive statistics were used to report means, standard deviations, and learning gain percentages across different age groups and genders. Effect sizes (Cohen's *d*) were also calculated to evaluate the practical significance of observed gains. Additionally, independent samples t-tests were conducted to assess gender differences.

Qualitative data were analyzed by two researchers using thematic coding, focusing on key themes such as learner engagement, perceived challenges, mentor support, and the development of problem-solving skills. Codes were derived

both deductively, based on the research questions, and inductively, from patterns emerging from participants' responses. To assess the reliability of coding, intercoder agreement was calculated using Cohen's *kappa.*

Game Development Activities

This game development activity centered on core computational thinking concepts such as loops and conditionals, applied through the creation of sprite-based games. Students programmed sprite movement using *"move," "if-then,"* and *"repeat until"* blocks to control the Pac-Man or Snake character. Collision detection was achieved using *"touching ular"* and *"if touching wall"* conditions to handle game-over scenarios.

The project emphasized the application of variables and algorithms. For example, students *created a "score"* variable that increased whenever the sprite collected an item (for example a dot in Pacman or apple/broccoli in Snake game), and used custom blocks to modularize functions such as resetting the game or increasing difficulty levels. These tasks were scaffolded through sample scripts, partial code templates, and in-class debugging exercises. Figure 1 illustrates the final Pacman and Snake game layout, highlighting how these elements were combined into a complete playable game.

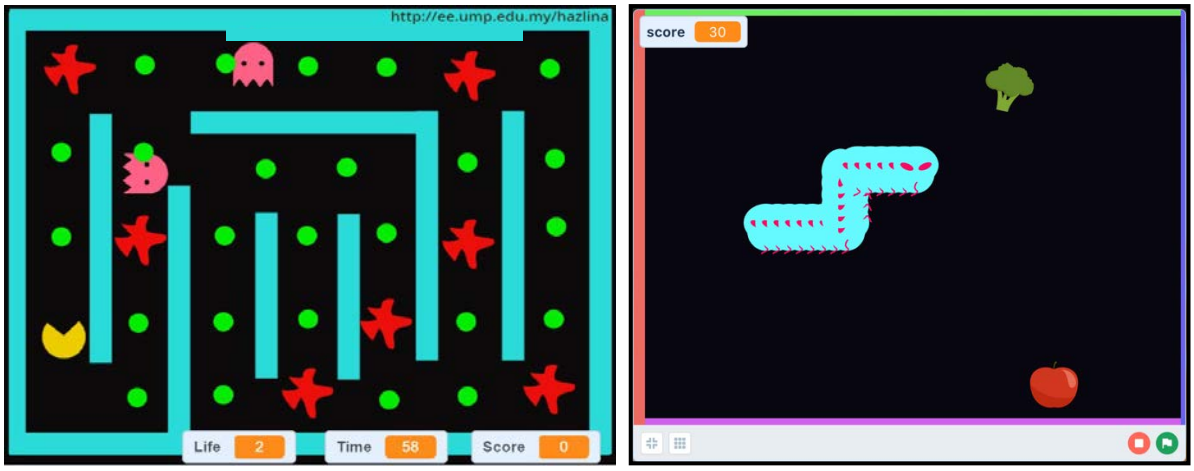

Figure 1. The Pacman and Snake Game final layout.

Instructional Stages

The instructional stages of this study are designed to incorporate tiered scaffolding, a pedagogical approach that provides varying levels of support to students based on their individual needs and progress (Noordin et al., 2024). This method was implemented through a structured sequence of programming tasks that gradually increased in complexity and autonomy.

In Stage 1 (Workout Programming), students began with fully guided exercises, where they followed step-by-step instructions to build basic sprite movement using blocks like "*move 10 steps*" and "*if key pressed*". Stage 2 (Debugging Specific Malfunctions) introduced semi-completed scripts containing common logical errors. Students were supported in identifying and correcting mistakes using "*if-else*", "*repeat*", and "*wait until*" blocks. Stage 3 (Semi-Completed Programming) involved providing learners with partially built games where they were required to add missing functionality, such as score tracking using variables and sprite interactions via collision detection blocks. In Stage 4 (New Programming Tasks), students worked independently to develop their own version of a Pac-Man or Snake game. Here, they applied all previously learned concepts to design full game logic, build interfaces, and implement win/lose conditions.

This scaffolding approach is applied in the context of programming tasks, debugging specific malfunctions, working with semi-completed programs, and tackling new programming tasks. This progression from high-support to low-support tasks reflects the Zone of Proximal Development (ZPD) principle, where learners are initially guided but are gradually empowered to problem-solve independently.

Tiered Scaffolding in Programming Education

Tiered scaffolding involves providing initial high levels of support, which are gradually reduced as students gain competence and confidence. This method aligns with Vygotsky's Zone of Proximal Development (ZPD), where learners can achieve higher levels of understanding with appropriate guidance (Shin et al., 2025).

In the context of this study, participants progressed through four scaffolding stages: workout programming, debugging malfunctions, completing semi-built games, and designing new games independently. During each stage, students engaged in hands-on coding tasks using mBlock. They experimented with commands, modified scripts, tested sprite interactions, and refined game logic based on feedback. Facilitators played a key role as instructor and mentors. They

conducted short live demonstrations, distributed scaffolded worksheets, and provided real-time feedback and troubleshooting support throughout the sessions. They also monitored group dynamics, encouraged peer collaboration, and ensured that all participants were progressing at their own pace. Tiered scaffolding is implemented through these structured stages, as illustrated in Figure 2, with the level of instructional support adjusted based on individual learner needs and readiness.

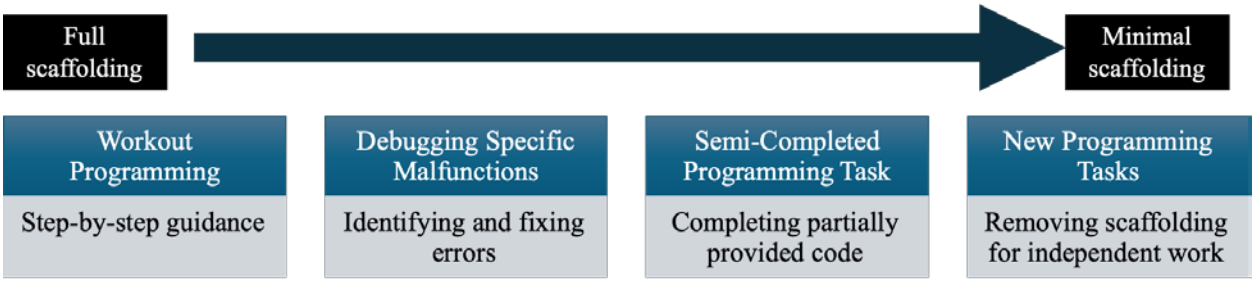

Figure 2 Tiered Scaffolding in Programming Education

During Workout Programming, students begin with detailed, fully guided exercises delivered via printed worksheets, and instructor demonstrations. These materials introduced fundamental programming concepts such as sprite movement and event handling. Instructors modeled each step on-screen while students followed along individually or in pairs. As students became more comfortable, researchers gradually reduced the step-by-step prompts, allowing learners to complete similar tasks with less guidance.

In Debugging Specific Malfunctions, students were given pre-built but partially complete mBlock projects. Initially, these included hints and correction scaffolds, such as labels or highlighted blocks indicating potential errors. Instructors and mentors facilitated one-on-one or small group troubleshooting, helping participants reflect on the logic behind each fix.. Over time, the debugging tasks became more complex issues with minimal external hints, strengthening their independent problem-solving abilities.

For Semi-Completed Programming, participants worked on partially written game projects that contained missing segments of logic, such as incomplete score tracking or movement blocks. With initial guidance, participants filled in the missing blocks by referencing prior examples and support materials. As scaffolding faded, participants were expected to complete more substantial portions of code independently, thereby reinforcing their understanding of program flow and structure.

Moving to New Programming Tasks, participants were challenged to create original features, such as a timer, sound effects, or increasing difficulty, or entirely new games using the concepts they had learned. Initially, they were provided with challenge cards, example templates, and structured checklists. As the final stage of scaffolding, instructors stepped back, allowing students to propose, design, and build solutions independently. Researchers observed, provided feedback when requested, and documented learning progression. This stage emphasized creativity, autonomy, and integration of multiple CT concepts.

The activities are divided into four instructional stages, as shown in Table 2, each designed to scaffold the learning process and progressively build students' CT skills. Tiered scaffolding was applied through planned support withdrawal, peer discussion, task variation, and facilitation, ensuring active involvement and varying levels of support based on participants' needs and progress.

Table 2. Instructional Stages in Game Development with Graphical Programming

Objective	Activities	Tiered Scaffolding
Stage 1 – Introduction to mBlock		
Familiarize students with the mBlock interface and basic programming blocks	A brief tutorial on navigating the mBlock environment, followed by simple exercises to practice using basic blocks	Initial exercises were fully guided, with step-by-step instructions. As students became more comfortable, they were given semi-completed programs to modify and complete independently. (2 hours)
Stage 2 – Game Decomposition		
Help students break down the game development process into manageable tasks.	Students analyzed the components of Pac-Man and Snake, identifying key elements such as sprites, movements, and interactions. They created flowcharts to visualize the game structure	Students started with detailed guidance on decomposing game elements. Gradually, they were encouraged to decompose new tasks with minimal assistance. (3 hours)
Stage 3 – Algorithm Design		
Guide students in structuring the game rules and logic.	Students designed algorithms for sprite movements, collision detection, and score tracking. They began with a pseudocode to outline their logic before implementing it in mBlock. Figure 3 illustrates a sample block-based solution created during this stage, demonstrating the use of loops, conditionals, and variables in controlling game mechanics.	Initially, students received templates and examples to follow. As their skills improved, they were tasked with designing algorithms for new game features independently. (4 hours)
Stage 4 – Debugging & Iteration		
Teach students to identify and fix errors in their code, and to enhance their games with additional features.	Students tested their games, identified bugs, and iteratively refined their code. They were encouraged to add new features, such as increasing difficulty levels or adding sound effects, to enhance their games.	Debugging started with guided identification of common errors. Over time, students were given more complex, semi-completed programs with specific malfunctions to debug on their own. (3 hours)

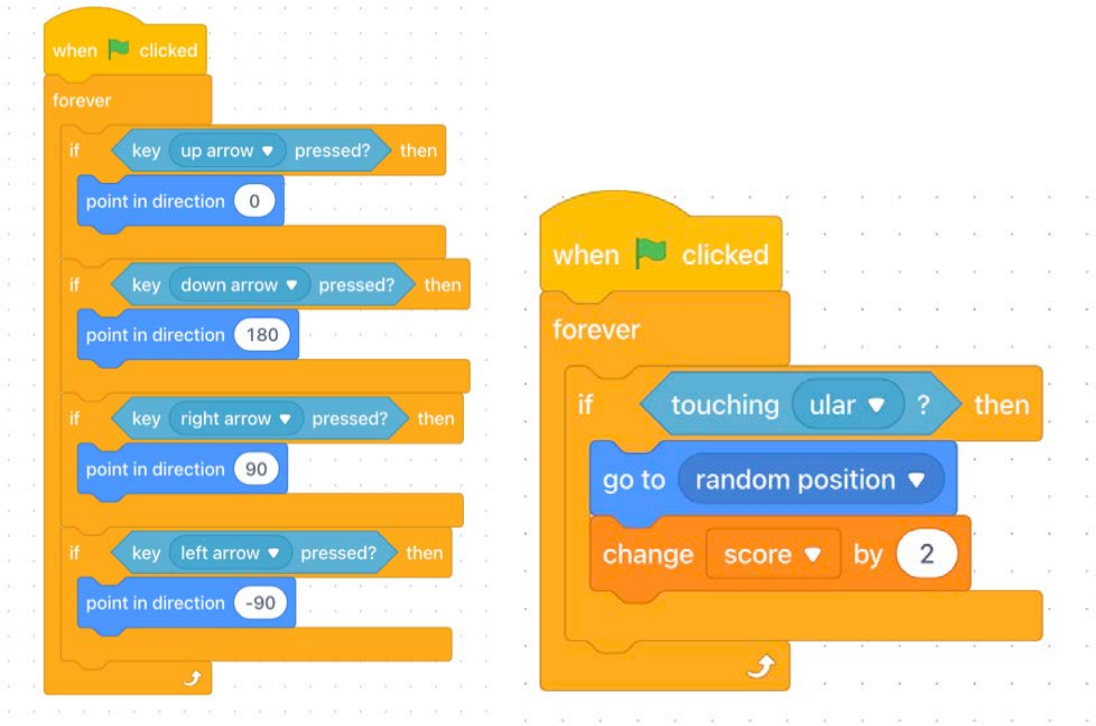

Figure 3. Block-Based Algorithm Design (Stage 3) for Snake Game in mBlock

Results

Pre-Test and Post-Test Performance

Following the scaffolded game development intervention, all participant groups demonstrated improvements in CT skills. Table 3 summarizes the mean pre-test and post-test scores across different age groups.

Table 3. Age Group Pre and Post Test Evaluation

Age_Group/ Score		POST (%)	PRE (%)	Learning Gain (POST - PRE%)
10 - 11 (n = 207)	Mean	45.1	36.4	8.67
	δ	11.7	14.5	9.17
	Min	13.8	0	-0.5
	Max	72.4	67.9	48.3
12 - 13 (n = 48)	Mean	42.6	34.3	8.3
	δ	12.3	13.4	8.25
	Min	17.2	3.6	0.7
	Max	70	67.9	33.5
14 - 15 (n = 37)	Mean	35.7	29.2	6.48
	δ	10.9	11.6	5.03
	Min	17.2	10.7	0.7
	Max	65.5	60.7	19.8
Teachers (n = 18)	Mean	48.9	38.4	10.5
	δ	10.5	12	7.03
	Min	34.5	14.3	1.7
	Max	66	60.7	26.8

δ = standard deviation

A paired sample t-test showed that the improvement from pre- to post-test scores was statistically significant across all groups, $t(309) = 15.24$, $p < .001$. The overall effect size was Cohen's $d = 0.65$, reflecting a moderate to strong impact of the intervention. When analyzed by age group, participants aged *10–11* demonstrated a large effect size ($d = 0.95$), while students aged 12–13 and 14–15 showed even greater learning gains, with effect sizes of $d = 1.01$ and $d = 1.29$, respectively. The strongest improvement was observed among the teacher group, who recorded a very large effect size of $d = 1.49$. These findings suggest that the tiered scaffolding model was highly effective in improving computational thinking skills, especially for both novice learners and adult participants.

Gender-Based Analysis

Both male and female participants exhibited statistically significant learning gains following the intervention, as shown in Table 4. Female students started with a higher pre-test mean score ($M = 40.49\%$, **δ** = 13.51), improving to a post-test mean of 47.40% (**δ** = 11.95), resulting in a mean gain of 6.91 percentage points ($t(112) = 11.09$, $p < .001$). Male participants began with a lower pre-test mean ($M = 32.43\%$, **δ** = 13.56) but demonstrated a larger learning gain of 9.34 points ($t(196) = 14.02$, $p < .001$), reaching a post-test mean of 41.78% (**δ** = 11.64).

A strong positive correlation was observed between pre- and post-test scores for both females ($r = .872$, $p < .001$) and males ($r = .734$, $p < .001$), indicating consistency in learning outcomes. While both groups benefited from the scaffolded intervention, the larger gains among male students suggest the needs for further exploration in relation to instructional preferences or initial familiarity with the content.

Table 4. Gender Pre Test and Post Test Evaluation

Gender/ Score	Female (n = 113)				Male (n = 197)			
	Mean	δ	Min	Max	Mean	δ	Min	Max
POST (%)	47.398	11.951	13.8	70	41.777	11.6441	17.2	72.4
PRE (%)	40.488	13.5118	10.7	67.9	32.433	13.5581	0	67.9
Learning Gain (POST - PRE%)	6.91	6.6211	-0.5	33.9	9.344	9.355	0.7	48.3

δ = standard deviation

Paired Samples Statistical Analysis

Correlation analysis revealed strong positive relationships between pre- and post-test scores for all groups ($r > 0.78$), as shown in Table 5, indicating consistent learning progression. The highest correlation was observed in the 14–15 age group ($r = 0.90$), which may reflect increased cognitive maturity or self-regulation among older students. Paired sample

t-tests indicated that the improvements from pre-test to post-test were statistically significant in all groups ($p < .001$). These results provide strong evidence for the consistency and reliability of the observed learning gains following the scaffolded game development intervention.

Table 5. Paired Sample Test Analysis

Age Group	Score (%)	Paired Samples Statistics			Correlation	Paired Differences					*t*	df
		Mean	δ	Std. Error Mean		Mean	δ	Std. Error Mean	95%Confidence Interval Lower	Upper		
10 - 11 (n = 207)	POST	45.12	11.72	0.81	0.78	8.67	9.17	0.64	7.41	9.93	13.61	206
	PRE	36.45	14.54	1.01								
12 - 13 (n = 48)	POST	42.60	12.26	1.77	0.80	8.30	8.25	1.19	5.91	10.70	6.97	47
	PRE	34.30	13.42	1.94								
14 - 15 (n = 37)	POST	35.72	10.89	1.79	0.90	6.48	5.03	0.83	4.80	8.15	7.84	36
	PRE	29.25	11.60	1.91								
Teachers (n = 18)	POST	48.89	10.55	2.49	0.81	10.49	7.03	1.66	6.99	13.98	6.33	17
	PRE	38.40	11.96	2.82								

δ = standard deviation

Students' Perceptions of the Learning Experience

Student perceptions were captured via a 7-item Likert-scale survey (A1–A7), as discussed in Table 1, which assessed aspects of programming understanding, skill development, and engagement. Table 6 shows disaggregated ratings by age and gender. The highest-rated items were A2 and A6, reflecting increased confidence in physical computing and perceived technical skill development. Female students in the 14–15 group gave consistently higher scores across all items, suggesting greater engagement with the scaffolded learning process. Overall mean scores exceeded 4.0 on a 5-point scale, indicating strong participant satisfaction with the program.

Table 6. Participant Feedback on Scaffolded Game Development Activities by Age Group and Gender

Age Group	Gender	Details	A1	A2	A3	A4	A5	A6	A7
10 – 11 (n = 207)	Female	Mean	4.06	4.19	4.08	4.1	3.81	4.14	4.08
		Median	4	4	4	4	4	4	4
		δ	1.125	1.014	1.13	1.008	1.167	0.984	1.095
		Variance	1.265	1.028	1.276	1.015	1.361	0.968	1.199
	Male	Mean	3.94	4.14	4.14	4.01	3.93	4.11	4.11
		Median	4	4	4	4	4	4	4
		δ	0.876	0.928	0.903	0.96	0.915	0.898	0.835
		Variance	0.768	0.862	0.815	0.921	0.838	0.807	0.697
12 – 13 (n = 48)	Female	Mean	4.29	4.29	4.43	4.29	4.5	4.57	4.36
		Median	4.5	4	4	4	4.5	5	5
		δ	1.069	0.825	0.514	0.469	0.519	0.514	1.082
		Variance	1.143	0.681	0.264	0.22	0.269	0.264	1.17
	Male	Mean	4.41	4.44	4.26	4.29	4.15	4.41	4.59
		Median	4	4	4	4	4	4	5
		δ	0.5	0.504	0.79	0.76	0.821	0.5	0.5
		Variance	0.25	0.254	0.625	0.578	0.675	0.25	0.25
14 – 15 (n = 37)	Female	Mean	4.58	4.5	4.33	4.42	4.33	4.58	4.67
		Median	5	4.5	4.5	5	4.5	5	5
		δ	0.515	0.522	0.888	0.9	0.888	0.515	0.492
		Variance	0.265	0.273	0.788	0.811	0.788	0.265	0.242
	Male	Mean	4.04	4.16	4.08	4.08	3.92	4.04	4.08
		Median	4	4	4	4	4	4	4
		δ	0.735	0.8	0.909	0.759	0.812	0.735	0.759
		Variance	0.54	0.64	0.827	0.577	0.66	0.54	0.577

Table 7. Continued

Age Group	Gender	Details	A1	A2	A3	A4	A5	A6	A7
Teachers (n = 18)	Female	Mean	4.14	4.43	4.43	4.43	4.29	4.29	4.29
		Median	4	4	4	4	4	4	4
		δ	1.069	0.535	0.535	0.535	0.488	0.488	0.488
		Variance	1.143	0.286	0.286	0.286	0.238	0.238	0.238
	Male	Mean	4.2	4	4.1	4	4	4.2	4.1
		Median	4	4	4	4	4	4	4
		δ	0.422	1.155	0.316	0.816	0.816	0.422	1.197
		Variance	0.178	1.333	0.1	0.667	0.667	0.178	1.433

δ = standard deviation

Qualitative Insights

Qualitative feedback obtained through 38 semi-structured interviews further supported the quantitative findings. Thematic analysis of student feedback highlighted four key themes, namely skill acquisition, mentor support, creativity and engagement, and career interest**.** Coding reliability was established through intercoder agreement, with a Cohen's *kappa* of 0.81, indicating strong consistency.

Skill acquisition was widely reported, with students describing improved understanding of programming concepts and technical abilities programming in mBlock. Many students reported that they "*learned how to make a game*", "*learned the programming blocks in mBlock*", "*I learned how to use mBlock and build games with sprite movement and score tracking*", or gained "*knowledge about coding and creating sprites*," reflecting development in core computational thinking skills. These statements demonstrate improved confidence and competence in using graphical programming tools.

Mentor support is an important factor that aligned with Vygotsky's Zone of Proximal Development. Participants acknowledging that guidance from facilitators helped them navigate challenges. Responses such as the mentors "*helped step by step,*" "*used easy language*," and "*helped me to understand the issues I'm facing*," suggest that scaffolded guidance played a key role in their progression from basic to more complex tasks. These reflections align with the scaffolding principles of timely support and personalized guidance.

Creativity and engagement were consistently noted, as students found the game development activities enjoyable and motivating. Comments such as "*making the snake teleport*," "*designing characters and environments*," and "*creating interactive games*" illustrated how participants were motivated by hands-on, goal-oriented projects. These responses reflect how the scaffolded, project-based approach fostered intrinsic motivation and enjoyment in learning.

The final theme of *career interest* revealed the program's broader influence. Several students noted their ambition to become an "*engineer,*" "*scientist,*" or "*roboticist*," while others stated, "*this program makes me want to learn more about programming*," or "*because this gives better career opportunities*." These responses indicate that scaffolded game development may contribute to STEM identity formation and interest in future study.

Discussion

This study evaluated the impact of scaffolded game development activities delivered through a graphical programming environment on students' CT skills. The findings demonstrate improvements in CT competencies across all age and gender groups, with particularly strong gains among younger students (10–11 years; $d = 0.95$) and teachers ($d = 1.49$). These results highlight the potential of structured, scaffolded programming tasks to enhance learners' understanding of key CT concepts, including loops, conditionals, debugging, and algorithmic thinking. The results align with and extend prior research while highlighting the effectiveness of integrating tiered scaffolding with creative, project-based learning. Consistent with Vygotsky's Zone of Proximal Development (ZPD), younger students (especially those aged 10–11) exhibited the most learning gains. This suggests that tiered scaffolding, where instructional support is gradually reduced, enables novice learners to successfully develop programming proficiency in a structured and manageable way. This finding is consistent with van de Pol et al. (2015), which demonstrated that tailoring scaffolding to learners' developmental levels enhances engagement and academic success (van de Pol et al., 2015).

The teacher group showed highest learning gains, even though many had limited programming experience prior to the intervention. This highlights the adaptability of scaffolded game development for diverse learner profiles, including adult learners. However, implementation may have varied depending on the teachers' comfort levels with the tools, a factor which could have influenced outcomes. This variability mirrors findings by Fernández et al. (2001) on the importance of instructor support in collaborative scaffolding environments (Fernández et al., 2001).

In terms of gender, the quantitative results showed that both male and female learners improved, with males recording slightly higher gains. However, females consistently rated the learning experience more positively, particularly in older age groups. This suggests that while graphical programming environments like mBlock are inclusive and accessible (Rao

& Bhagat, 2024), factors such as prior exposure, self-efficacy, or classroom dynamics may influence performance outcomes in which a pattern that merits deeper investigation.

The qualitative findings complement and contextualize the quantitative data. Students frequently expressed that the scaffolded activities increased their interest and confidence in programming, particularly due to the opportunity to create tangible game outcomes. The project-based nature of game design made abstract CT concepts more relatable, aligning with Wing's (Wing, 2006, 2017) call for integrating CT into real-world problem-solving. The most frequently cited enablers of success were mentor guidance and the ability to engage in iterative, hands-on learning, in which are the key characteristics of scaffolded instruction.

The research implements a structured scaffolding framework, leading up to a final design task involving the development of a complete Pac-Man or Snake game. This approach provided students with an end-to-end learning arc, requiring them to apply decomposition, abstraction, and algorithmic thinking in an integrated way. Such application of CT aligns with constructionist theories that emphasize learning-by-doing.

Conclusion

This study demonstrates that scaffolded game development activities, delivered through a graphical programming environment, has a positive influence on computational thinking (CT) skills among young learners. By integrating Vygotsky's scaffolding theory with Wing's computational thinking framework, the intervention provided structured, engaging pathways for students to build core programming competencies. Quantitative improvements across all participant groups, alongside positive qualitative feedback, further supports the effectiveness of using tiered, scaffolded learning approaches in early CT education. Beyond technical skill development, the scaffolded activities encourage creativity, critical thinking, and greater confidence in problem-solving, which is a key competency needed in today's digital society. The results suggest that such approaches are both inclusive and adaptable, offering accessible entry points into programming education for diverse learner populations, regardless of prior experience. Moving forward, future research could explore the long-term impact of scaffolded game development interventions, particularly in tracking the retention and transferability of CT skills across subjects. Integrating emerging technologies such as AI-assisted scaffolding or augmented reality could further personalize the learning experience and enhance scalability across broader educational contexts. In addition, comparative studies between rural and urban school settings could shed light on context-specific implementation outcomes. By equipping students with foundational CT skills through structured yet creative learning experiences, scaffolded game development offers a promising pathway toward preparing young learners for lifelong adaptability and success in an increasingly digital world.

Recommendations

Building on these findings, several recommendations are proposed for future research and practice. A potential way forward for this study includes expanding the participant pool to include more diverse demographics, encompassing different educational backgrounds, age groups, and regions, to better validate the scalability and adaptability of the scaffolded game development model. Another important direction for expanding this research is to conduct longitudinal studies that examine the long-term retention of computational thinking skills and their transferability beyond programming tasks, including applications in mathematics, science, and creative problem-solving contexts. On the other hand, integrating scaffolded graphical programming with gradual introductions to text-based coding could help bridge the gap between visual and traditional programming environments. The exploration of emerging technologies, such as AI-assisted scaffolding or augmented reality environments, could offer new opportunities for personalizing learning paths and enhancing engagement, making computational thinking education more accessible and sustainable across diverse educational settings.

Limitations

While this study demonstrates the effectiveness of scaffolded game development activities in enhancing CT skills among young learners, several limitations must be acknowledged. The study was conducted with participants from a specific age range (10–15 years) and geographical region, which may limit the generalizability of the findings to broader or more diverse populations. The intervention of this work primarily focused on beginner-level graphical programming using mBlock and thus may not fully capture challenges associated with transitioning to more complex or text-based programming environments. The assessment in this work focused on short-term learning gains immediately following the intervention. Longer-term retention and the transferability of CT skills across different subjects or real-world contexts were not evaluated. While qualitative feedback provided valuable insights, the use of more systematic qualitative methods such as extended interviews or classroom observations could have yielded deeper understanding of the learning processes and experiences.

Ethics Statements

The study was conducted in adherence to ethical guidelines and approved under the IIUM Research Ethics Committee protocol (IREC2023-173), ensuring the rights, privacy, and well-being of all participants were safeguarded throughout the research process.

Acknowledgements

The author would like to express sincere gratitude to all participants, facilitators and student mentors who has involved in the UMPSA STEM Lab activities. Their enthusiastic participation, valuable feedback, and support have greatly contributed to the success of this research.

Conflict of Interest

The authors have no competing interests to declare that are relevant to the content of this article.

Funding

This work has been supported by the UIC231519 / RDU232710 grant, Universiti Malaysia Pahang Al-Sultan Abdullah (UMPSA), IBM Malaysia and the Ministry of Higher Education Malaysia (MoHE). Their involvement enabled the successful completion of this research at the UMPSA STEM Lab, Faculty of Electrical and Electronics Engineering Technology, UMPSA.

Generative AI Statement

Generative AI tools were used solely to enhance language clarity and improve the readability of the manuscript. No AI-generated content, data, or analysis was included in the research findings. The final version of the manuscript was thoroughly reviewed and verified to ensure accuracy and originality.

References

Bachiller-Burgos, P., Barbecho, I., Calderita, L. V., Bustos, P., & Manso, L. J. (2020). LearnBlock: A robot-agnostic educational programming tool. *IEEE Access*, *8*, 30012-30026. https://doi.org/10.1109/ACCESS.2020.2972410

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 Annual Meeting of the American Educational Research Association*, (pp. 1-25). American Educational Research Association. http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Dilmen, K., Kert, S. B., & Uğraş, T. (2023). Children's coding experiences in a block-based coding environment: A usability study on code.org. *Education and Information Technologies*, *28*, 10839-10864. https://doi.org/10.1007/s10639-023-11625-8

Fernández, M., Wegerif, R., Mercer, N., & Rojas-Drummond, S. (2001). Re-conceptualizing" scaffolding" and the zone of proximal development in the context of symmetrical collaborative learning. *The Journal of Classroom Interaction*, *36/37*(2/1), 40-54. https://www.jstor.org/stable/23869224

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher*, *42*(1), 38-43. https://doi.org/10.3102/0013189X12463051

Lodi, M., & Martini, S. (2021). Computational thinking, between papert and wing. *Science and Education*, *30*(4), 883-908. https://doi.org/10.1007/s11191-021-00202-5

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, *41*, 51-61. https://doi.org/10.1016/j.chb.2014.09.012

Margolis, A. A. (2020). Zone of Proximal development, scaffolding and teaching practice. *Cultural-Historical Psychology*, *16*(3), 15-26. https://doi.org/10.17759/chp.2020160303

Noordin, N. H., Abdullah, K. K. B. H., & Eu, P. S. (2024). Assessing the Effectiveness of UMP STEM Cube as a Tool for Developing Digital Making Skill Sets. *IEEE Transactions on Education*, *67*(6), 857-867. https://doi.org/10.1109/TE.2024.3376448

Rao, T. S. S., & Bhagat, K. K. (2024). Computational thinking for the digital age: A systematic review of tools, pedagogical strategies, and assessment practices. *Educational Technology Research and Development*, *72*, 1893-1924. https://doi.org/10.1007/s11423-024-10364-y

Shabani, K., Khatib, M., & Ebadi, S. (2010). Vygotsky's zone of proximal development: Instructional implications and teachers' professional development. *English Language Teaching*, *3*(4), 237-248. https://doi.org/10.5539/elt.v3n4p237

Shin, Y., Jung, J., Choi, S., & Jung, B. (2025). The influence of scaffolding for computational thinking on cognitive load and problem-solving skills in collaborative programming. *Education and Information Technologies*, *30*, 583-606. https://doi.org/10.1007/s10639-024-13104-0

Srisangngam, P., & Dechsura, C. (2020). STEM education activities development to promote computational thinking's Students. *2020 5th International STEM Education Conference (ISTEM-Ed)* (pp. 103-105). https://doi.org/10.1109/iSTEM-Ed50324.2020.9332734

Sun, D., Looi, C.-K., Li, Y., Zhu, C., Zhu, C., & Cheng, M. (2024). Block-based versus text-based programming: A comparison of learners' programming behaviors, computational thinking skills and attitudes toward programming. *Educational Technology Research and Development*, *72*, 1067-1089. https://doi.org/10.1007/s11423-023-10328-8

van de Pol, J., Volman, M., Oort, F., & Beishuizen, J. (2015). The effects of scaffolding in the classroom: Support contingency and student independent working time in relation to student achievement, task effort and appreciation of support. *Instructional Science*, *43*, 615-641. https://doi.org/10.1007/s11251-015-9351-z

Wan, H., Zhang, X., Yang, X., & Li, S. (2024). Which approach is effective: Comparing problematization-oriented and structuring-oriented scaffolding in instructional videos for programming education. *Education and Information Technologies*, *29*, 17807-17823. https://doi.org/10.1007/s10639-024-12550-0

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2017). Computational thinking's influence on research and education for all. *Italian Journal of Educational Technology*, *25*(2), 7-14. https://doi.org/10.17471/2499-4324/922

Ye, H., Liang, B., Ng, O.-L., & Chai, C. S. (2023). Integration of computational thinking in K-12 mathematics education: A systematic review on CT-based mathematics instruction and student learning. *International Journal of STEM Education*, *10*, Article 3. https://doi.org/10.1186/s40594-023-00396-w